

Sending SMS using Skype

Skype's API and Chat Command as Short Message Service

by Bob Swart

In the August 2005 issue of The Delphi Magazine, I covered the usage of the Skype API from within a Delphi application. One of the example applications I've designed and implemented in that issue was a Skype answering machine. It has been running for a while on my personal web server, but is now disconnected again (frankly, it took too many resources, and the quality of the incoming voice messages was too low to make it worthwhile to continue anyway).

Skype 2.0 introduced some new features to Skype, including video calling. This is not an article about video calling, however, but about another feature which has been available in Skype for some time: chatting. The Skype API protocol level 5 and later support the functionality to start a chat session manually (without using the Skype client itself). This article explains how to use the Skype API to start a chat, send a message, and end a chat, combining these commands into a simulated Skype SMS application.

Skype API

If you don't know what Skype is or how to use it, I recommend a visit to <http://www.drBob42.com/Skype> and the article I wrote for The Delphi Magazine issue #120 (see <http://www.TheDelphiMagazine.com>). I won't repeat how to download Skype or how to use the Skype API in a Delphi application – please either download my example source code or read the article for more details.

For the example in this article, I assume you have the Skype client version 1.3 or later installed on your machine (the minimum Skype API protocol level that supports chats is protocol 5, which if I remember correctly was the level of Skype version 1.3, or perhaps even earlier versions). Most people will run a newer Skype client nowadays, and although it's recommended to check for the supported protocol level, I leave that as an exercise for the reader.

Basically, in order to produce a Skype plug-in application, we need to set up a communication between our application and the Skype client. We first need to register two Skype client specific Windows messages, called SkypeControlAPIDiscover and SkypeControlAPIAttach.

```
procedure TfrmSkype.FormCreate(Sender: TObject);
begin
  WM_SkypeControlAPIDiscover := RegisterWindowMessage('SkypeControlAPIDiscover');
  WM_SkypeControlAPIAttach := RegisterWindowMessage('SkypeControlAPIAttach');
  SendMessage(HWND_BROADCAST, WM_SkypeControlAPIDiscover, Handle, 0)
end;
```

In response to the WM_SkypeControlAPIDiscover message, the Skype Client – when active – will respond with a WM_SkypeControlAPIAttach message, holding the SkypeAPIWindowHandle in the WParam part of that message.

```
procedure TfrmSkype.WndProc(var Message: TMessage);
begin
  if Message.Msg = WM_SkypeControlAPIAttach then
  begin
    if Message.LParam = 0 then
      HWND_SkypeAPIWindowHandle := Message.WParam;
    Message.Result := 1
  end
  else
    inherited
end;
```

Once we have the Skype Client Window Handle, we can send Skype API commands to it using the WM_CopyData message, which can be done with the following method:

```

procedure TfrmSkype.SendCommandToSkype(Str: String);
var
  CopyData: CopyDataStruct;
begin
  if Str <> '' then
  begin
    CopyData.dwData := 0;
    CopyData.lpData := PChar(Str);
    CopyData.cbData := Length(Str)+1;
    SendMessage(HWND_SkypeAPIWindowHandle, WM_COPYDATA, Handle, LPARAM(@CopyData))
  end
end;

```

Skype Chat Command

The Skype API consists of a number of commands that can be sent to the Skype Client window using the WM_CopyData windows message. As of protocol level 5, the Skype API is extended with support for Skype chats. Skype users are identified by their Skype handle (I have Skype handles eBob42 and drbob42 for example). In order to start a Skype chat with Skype user drbob42, you need to send the following command:

```
CHAT CREATE drbob42
```

If the specified user with Skype handle drbob42 can be reached (even if he or she is “away” or “not available” or has the status “do not disturb”, but not if the Skype user is not online), then the Skype client will respond with the following:

```
CHAT id STATUS DIALOG
```

ID is the unique chat id that is given to this new chat. We must use this chat id for the remainder of the chat, for example to send chat messages or close the chat when we’re done (otherwise the chat will close automatically if the user disconnects). The format of the id doesn’t matter at this time, we just need to obtain the id and use it from now on.

The code to extract the id from the incoming message can be found in the following listing, which is the WMCopyData method:

```

procedure TfrmSkype.WMCopyData(var Message: TWMCopyData);
var
  ID: String;
begin
  if (Message.From = HWND_SkypeAPIWindowHandle) and
    (HWND_SkypeAPIWindowHandle > 0) then
  begin
    ID := PChar(Message.CopyDataStruct.lpData);
    if Pos('CHAT', ID) = 1 then
    begin
      Delete(ID,1,5);
      if Pos('STATUS DIALOG', ID) > 0 then
      begin
        Delete(ID,Pos('STATUS DIALOG', ID),255);
        SendCommandToSkype('CHATMESSAGE ' + ID + ChatMessage)
      end
    end;
    Message.Result := 1
  end
end;

```

As soon as we have the chat id, we can send one or more messages to that id, with the following command:

```
CHATMESSAGE id ChatMessage
```

ChatMessage is the actual message you want to send (can be multiple lines of text if you wish).

Finally, when you want to end a chat, you can send the following command to the Skype Client:

```
ALTER CHAT id LEAVE
```

This will ensure that the receiving end will get the message that you’ve left the chat.

Simulating SMS using Skype Chat

SMS, or Short Message Service, consists of the ability to send a short text message to a mobile phone. In chat building blocks, sending an SMS can be simulated by starting a chat, sending a single message, and then leaving the chat right away. As long as the user you want to send the message to can be reached (i.e. is not offline), the chat message will be received. When that user has the status “do not disturb”, then the incoming message will not be shown on screen, but in the event log of the Skype Client. Otherwise, the message will appear on screen right away.

Using this knowledge, I've built a simple Skype single message service unit, with one exported procedure defined in the interface section as follows:

```
procedure SendSkypeSMS(const Handle: String = ''; const Message: String = '';
  AutoSend: Boolean = False);
```

Note that all arguments have default values.

The implementation of the unit uses a hidden form, and contains the following code:

```
implementation
uses
  Windows, Messages, Forms, StdCtrls, Registry;

{$R *.dfm}

type
  TfrmSkype = class(TForm)
    edMessage: TEdit;
    edID: TEdit;
    lbHandle: TLabel;
    lbMessage: TLabel;
    btnSMS: TButton;
    procedure FormCreate(Sender: TObject);
    procedure btnSMSClick(Sender: TObject);
  private
    WM_SkypeControlAPIDiscover: DWord;
    WM_SkypeControlAPIAttach: DWord;
    HWND_SkypeAPIWindowHandle: Cardinal;
  protected
    procedure SendCommandToSkype(Str: String);
    procedure WndProc(var Message: TMessage); override;
    procedure WMCopyData(var Message: TWMCopyData); message WM_COPYDATA;
  private
    ChatID: String;
  end;

var
  frmSkype: TfrmSkype = nil;

function SkypeClientLocation: String;
var
  Reg: TRegistry;
begin
  Result := '';
  Reg := TRegistry.Create(KEY_READ);
  try
    Reg.RootKey := HKEY_LOCAL_MACHINE;
    if Reg.OpenKey('SOFTWARE\Skype\Phone', False) then
      Result := Reg.ReadString('SkypePath');
  finally
    Reg.Free;
  end;
end;

procedure SendSkypeSMS(const Handle: String = ''; const Message: String = '';
  AutoSend: Boolean = False);
begin
  if SkypeClientLocation <> '' then
  begin
    if not Assigned(frmSkype) then
      frmSkype := TfrmSkype.Create(Application);
    frmSkype.ChatID := '';
```

```

    frmSkype.edID.Text := Handle;
    frmSkype.edMessage.Text := Message;
    if (Handle <> '') and (Message <> '') and AutoSend then
        frmSkype.SendCommandToSkype('CHAT CREATE ' + Handle)
    else
        frmSkype.ShowModal
    end
end
else
    MessageBox(Application.Handle,
        'The Skype client is not installed (or not found).'#13#10 +
        'Please download Skype from http://www.skype.com',
        'Dr.Bob''s Skype SMS Sender', MB_OK);
end;

procedure TfrmSkype.FormCreate(Sender: TObject);
begin
    WM_SkypeControlAPIDiscover := RegisterWindowMessage('SkypeControlAPIDiscover');
    WM_SkypeControlAPIAttach := RegisterWindowMessage('SkypeControlAPIAttach');
    SendMessage(HWND_BROADCAST, WM_SkypeControlAPIDiscover, Handle, 0)
end;

procedure TfrmSkype.WndProc(var Message: TMessage);
begin
    if Message.Msg = WM_SkypeControlAPIAttach then
        begin
            if Message.LParam = 0 then
                HWND_SkypeAPIWindowHandle := Message.WParam;
                Message.Result := 1
            end
        else
            inherited
        end;
end;

procedure TfrmSkype.SendCommandToSkype(Str: String);
var
    CopyData: CopyDataStruct;
begin
    if Str <> '' then
        begin
            CopyData.dwData := 0;
            CopyData.lpData := PChar(Str);
            CopyData.cbData := Length(Str)+1;
            SendMessage(HWND_SkypeAPIWindowHandle, WM_COPYDATA, Handle, LPARAM(@CopyData))
        end
    end;
end;

procedure TfrmSkype.WMCopyData(var Message: TWMCopyData);
var
    ID: String;
begin
    if (Message.From = HWND_SkypeAPIWindowHandle) and
        (HWND_SkypeAPIWindowHandle > 0) then
        begin
            ID := PChar(Message.CopyDataStruct.lpData);
            if Pos('CHAT', ID) = 1 then
                begin
                    Delete(ID,1,5);
                    if Pos('STATUS DIALOG', ID) > 0 then
                        begin
                            Delete(ID,Pos('STATUS DIALOG', ID),255);
                            if (edMessage.Text <> '') and (ChatID <> ID) then
                                begin
                                    SendCommandToSkype('CHATMESSAGE ' + ID + edMessage.Text);
                                    btnSMS.Enabled := True
                                end
                            else
                                SendCommandToSkype('ALTER CHAT ' + ID + 'LEAVE');
                                ChatID := ID
                            end
                        end;
                    Message.Result := 1
                end
            end;
        end;
end;
end;

```

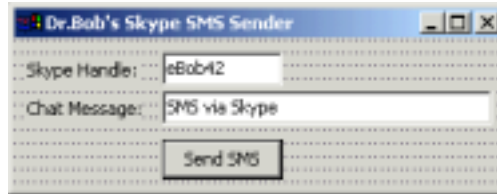
```

procedure TFrmSkype.btnSMSClick(Sender: TObject);
begin
    ChatID := '';
    btnSMS.Enabled := False;
    SendCommandToSkype('CHAT CREATE ' + edID.Text)
end;

end.

```

Note that the TFrmSkype is used to register the two Skype Windows messages, and for sending and receiving the WM_CopyData message. The form itself is designed as follows (at design-time):



Although this form is designed to allow the user to enter the Skype handle and message, this form will only be shown when needed. If you pass a value for the Skype Handle and Message, and pass True for the AutoSend parameter, then the Skype “SMS” message will automatically be sent, as demonstrated in the following example:

```

program SkypeSMS;
uses
    eBob42.SkypeSMS;
begin
    SendSkypeSMS('eBob42', 'Skype SMS is fun!', True)
end.

```

If, however, you call the SendSkypeSMS method without a Skype ID or message, and without passing True as third argument, then the dialog will appear, which allows you to specify the handle and message yourself before you can click on the Send SMS button. This allows for flexibility as well as direct Skype SMS services when needed.

Full source code of this example and other Skype plug-ins and applications can be downloaded from <http://www.drBob42.com/Skype>



Bob Swart b.swart@chello.nl (aka Dr Bob - www.drBob42.net) is a software developer, author, trainer, consultant and webmaster for his own one-man company, Bob Swart Training & Consultancy in Helmond, The Netherlands. He writes for numerous computing magazines as well as his own training material, and is also webmaster to the Developers Group.